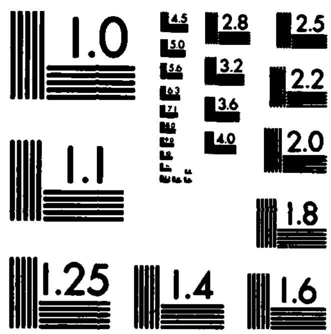
AD-A121 160 FINDING A MAJORITY AMONG N VOTES(U) YALE UNIV NEW HAVEN CT DEPT OF COMPUTER SCIENCE M J FISCHER ET AL. OCT 82 RR-252 N00014-80-C-0221 1/1

UNCLASSIFIED F/G 12/1 NL

END
FILMED
DTIC



(12)

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

FINDING A MAJORITY AMONG N VOTES

by
Michael J. Fischer and Steven L. Salzberg

Research Report # 252
October, 1982

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER 252	2. GOVT ACCESSION NO. A121760
	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) FINDING A MAJORITY AMONG N VOTES	5. TYPE OF REPORT & PERIOD COVERED Technical Report
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Michael J. Fischer and Steven L. Salzberg	8. CONTRACT OR GRANT NUMBER(s) ONR: N00014-80-0221 and NSF: MCS81-16678
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science/Yale University, Dunham Lab./10 Hillhouse Avenue, New Haven, Connecticut 06520	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR 049-456/11-5-81 410
11. CONTROLLING OFFICE NAME AND ADDRESS NSF, Washington, D.C. 20550/ Office of Naval Research, 800 N. Quincy, Arlington, VA 22217	12. REPORT DATE October, 1982
	13. NUMBER OF PAGES 9
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Office of Naval Research, 800 N. Quincy, Arlington, VA 22217	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distributed unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Voting, majority, analysis of algorithms, number of comparisons

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A commonly-used technique for fault-tolerant computing is to perform n redundant computations and then vote on the results, choosing on the majority value if one exists. We present an algorithm for carrying out the voting which uses $\lceil 3n/2 \rceil -2$ comparisons, and we prove the algorithms optimal. This solves Problem 81-5 posed in the <u>Journal of Algorithms</u>, June 1981.

DD FORM 1473, 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

FINDING A MAJORITY AMONG N VOTES

Solution to Problem 81-5 (Journal of Algorithms, June 1981)

Michael J. Fischer and Steven L. Salzberg

Department of Computer Science
Yale University
P.O. Box 2158 Yale Station
New Haven, Connecticut 06520

Manuscript Date
November 1981

Report Date
October 1982

A

*This work was supported in part by the Office of Naval Research under Contract N00014-80-C-0221 through a subcontract arrangement with the University of Washington, and by the National Science Foundation under Grant MCS81-16678.

FINDING A MAJORITY AMONG N VOTES
SOLUTION TO PROBLEM 81-5 (JOURNAL OF ALGORITHMS, JUNE 1981)

MICHAEL J. FISCHER AND STEVEN L. SALZBERG (YALE UNIVERSITY)

1. The Problem

The problem is as follows: given "a list of n numbers, representing the 'votes' of n processors on the result of some computation, we wish to decide if there is a majority vote and what that vote is. By majority vote we mean that more than half of the processors agree on the result of the computation. With how many comparisons among our n numbers can we solve this problem?" We present an algorithm followed by a proof of its optimality.

2. An Algorithm (Steven L. Salzberg)

The following algorithm gives the answer in at most ($\lceil 3n/2 \rceil - 2$) comparisons. Restate the problem as n balls, each of which is some color, and we want to find one ball representative of the majority color, if such a majority exists.

Observation

Suppose we arrange the balls so that no two adjacent balls are the same color. Then at most half (rounded up) of the balls on the list are the same color.

Algorithm

Phase 1: Take the balls one at a time and place them either on a list or in a "bucket." If the current ball is NOT the same color as the last ball on the list, then add the current ball to the list, and then, if the bucket is not empty, remove one ball from the bucket and place it also on the list. If it IS the same, place it in the bucket.

Phase 2: Use T for all comparisons in this phase, where T is the last ball on the list at the end of Phase 1. Repeatedly compare the current last ball on the list against T. If the comparison is EQUAL, throw the last two balls on the list away, unless only one ball remains on the list, in which case put it in the bucket instead. If the comparison is UNEQUAL, throw it and one ball from the bucket away. Continue in this way until the list is empty. During this process, if a ball is ever needed from the bucket and none is available, then halt and announce that no majority exists. When done, if the bucket is non-empty, announce T as representative of a majority. Otherwise, announce that no majority exists.

(Note: For efficiency, the algorithm can immediately halt in Phase 2 if n is even and the bucket ever becomes empty, since no majority would then be possible. However, this does not improve the worst-case behavior.)

Theorem 1: The algorithm above solves the majority balls problem and never uses more that $\lceil 3n/2 \rceil - 2$ comparisons.

Proof:

Correctness

At any stage during Phase 1, all the balls in the bucket (if any) are the same color as the last ball on the list. This property is guaranteed because whenever we add something to the list, we take something out of the bucket (without any comparison) and add it to the list as well. At the end of this phase, by the initial "observation", if there is a majority color, it must be the same color as T.

Phase 2 checks whether indeed a majority exists. Whenever a pair of balls is discarded, one is the same color as T and the other is different. Hence, T is a majority element iff a majority of the balls remaining at the end share its color. There are two cases. If Phase 2 terminates prematurely because a ball is needed from the bucket and the bucket is empty, then at most half the balls remaining on the list have color T; hence there is no majority. If the phase runs to completion, then all the remaining balls (if any) are in the bucket and have the same color as T. Hence, T represents a majority iff the bucket is non-empty.

Complexity

In Phase 1, the algorithm does $(n - 1)$ comparisons. In Phase 2, it makes one comparison for each pair of balls discarded after the first. In addition, it may make one comparison at the end which results in a ball being placed in the bucket instead of a pair being discarded. A straightforward case analysis shows the maximum number of compares for Phase 2 is $\lceil n/2 \rceil - 1$. Altogether then, the algorithm uses at most $\lceil 3n/2 \rceil - 2$ comparisons.

3. Optimality (Michael J. Fischer)

We construct an adversary which forces at least $2 * \lceil n/2 \rceil - 2$ unequal comparisons and at least $\lfloor n/2 \rfloor$ equal comparisons for a total of $\lceil 3n/2 \rceil - 2$.

The adversary maintains a partition of elements into two sets, the <u>arena</u> and the <u>outfield</u>. The arena contains a number of connected components of two types: "bars" and "flocks". A <u>bar</u> is a pair of elements with one unequal comparison between them. A <u>flock</u> is a non-empty set of elements connected by equal comparisons. Thus, a flock of k elements has at least k-1 equal comparisons among its members. Initially, each element is in a singleton flock.

At any stage in the algorithm, let B (resp. F) denote the number of bars (flocks) in the arena. Let t be the number of elements in the outfield, and let f be the total number of elements in all the flocks. Finally, let $m = \lfloor n/2 \rfloor + 1$ be the "majority number".

The adversary answers a question x:y of the algorithm as follows:

1. If x or y is in the outfield, the answer is "unequal".

2. If x (resp. y) is an element of a bar, the answer is "unequal", and x (resp. y) is moved to the outfield. The remaining element of the bar becomes a new singleton flock.

3. If x and y are both members of the same flock, the answer is "equal".

4. If x and y are in separate flocks, then there are two cases depending on $d = B + f$.

   Case 1) $d > m$: Then it will follow that both x and y are in singleton flocks, so the answer is "unequal", and $\{x,y\}$ becomes a new bar.

   Case 2) $d = m$: Then the answer is "equal", and the flocks containing x and y are merged together.

Note: Case 1 decreases d by 1 and Case 2 leaves it unchanged.

Claim 1: $d \geq m$. Moreover, if $d > m$, then all flocks are singletons.

Claim 2: At any time, the following two colorings are both consistent with all of the answers given by the adversary:

1. All elements are given distinct colors except that elements within the same flock are colored the same.

2. A single target color is assigned to all of the elements in all of the flocks and the same color is assigned to one element of each bar. The remaining elements each receive a distinct color.

Claim 3: No correct algorithm can stop until the arena contains only a single component, which will be a flock of size m.

Proof: Assume the arena contains two or more components. Then $n \geq 2$, so also $m \geq 2$. By the definition of d, each flock is strictly smaller than d. Every flock is also strictly smaller than m, for either $d = m$, or every flock is a singleton by Claim 1. Thus, the first coloring of Claim 2 fails to have a majority element. On the other hand, since $d \geq m$, the target color of the second coloring of Claim 2 is a majority. Since both colorings are possible, no correct algorithm can stop at this time. Hence, at termination there can be only one component, which must be a flock of size $d = m$ (by definition of d and Claim 1).

Claim 4: The number of unequal comparisons made by the algorithm at any stage is at least $2*t + B$, and the number of equal comparisons is at least $f - F$.

Proof: Easy induction.

Theorem 2: Consider any algorithm which solves the majority balls problem. Then there is an input on which it makes at least $2*(n - m) = 2*\lceil n/2 \rceil - 2$ unequal comparisons and at least $m - 1 = \lfloor n/2 \rfloor$ equal comparisons. Thus, the total number of comparisons is at least $\lceil 3n/2 \rceil - 2$.

Proof: By Claim 3, the arena contains a single component at termination which is a flock of exactly m elements. Hence, $t = n - m$, $B = 0$, $f = m$, and $F = 1$. The theorem follows immediately from Claim 4.

Acknowledgement

We thank David Lichtenstein for many helpful suggestions and discussions.

DISTRIBUTION LIST

Office of Naval Research Contract N00014-82-K-0154
Michael J. Fischer, Principal Investigator

Defense Technical Information Center
Building 5, Cameron Station
Alexandria, VA 22314
(12 copies)

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Dr. R.B. Grafton, Scientific Officer (1 copy)
Information Systems Program (437) (2 copies)
Code 200 (1 copy)
Code 455 (1 copy)
Code 458 (1 copy)

Office of Naval Research
Branch Office, Pasadena
1030 East Green Street
Pasadena, CA 91106
(1 copy)

Naval Research Laboratory
Technical Information Division
Code 2627
Washington, D.C. 20375
(6 copies)

Office of Naval Research
Resident Representative
715 Broadway, 5th floor
New York, N.Y. 10003
(1 copy)

Dr. A.L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, D.C. 20380
(1 copy)

Naval Ocean Systems Center
Advanced Software Technology Division
Code 5200
San Diego, CA 92152
(1 copy)

Mr. E.H. Gleissner
Naval Ship Research and Development Center
Computation and Mathematics Department
Bethesda, MD 20084
(1 copy)

Captain Grace M. Hopper (008)
Naval Data Automation Command
Washington Navy Yard
Building 166
Washington, D.C. 20374
(1 copy)

Defense Advance Research Projects Agency
ATTN: Program Management/MIS
1400 Wilson Boulevard
Arlington, VA 22209
(3 copies)